18.408: Algorithmic Aspects of Machine Learning

Lecture #1: Introduction

(on board before class)

Instructor: Ankur Moitra (moitra@mit.edu)

Lectures: Tu Th 11-12:30

Office Hours: TBA

Prereqs: 6.046 or equiv
18.600/6.041 or equiv
18.06/18.C06 or equiv

Assessment: 2-3 problem sets and research oriented final project

Website: people.csail.mit.edu/moitra/408b.html

Textbook: On website

Goal: Introduce you to major themes, via example

Poll: How many undergrads? grads? postdocs? other?

How many 18s? 6s? other?

This class: exploring the space btwn theoretical CS and ML

Models $\longleftrightarrow$ Algorithms

relationship is subtle

① Many expressive models for describing world around us

e.g. mixture models
graphical models
markov decision processes
linear dynamical systems
etc

But a model is only as good as our ability to use it!

② What about algorithms?

usually want them to work in worst-case

But computational intractability is everywhere, esp. in ML

③ So what can we do?

heuristics: seem to work, but when and why?

can we diagnose and improve them?

Main Q: what can models and algorithms teach us about each other?

Today: Nonnegative Matrix Factorization (NMF)

Let's start with Singular Value Decomp (SVD):

Given $m \times n$ matrix M, can write

$$M = U \Sigma V^T$$

where U, V are orthonormal, Σ is diagonal and nonnegative

Alternatively $M = \sum_{i=1}^{r} \sigma_i u_i v_i^T$

where u_i / v_i are i^{th} coln of U/V and σ_i is i^{th} diagonal entry of Σ

Fact: $\text{rank}(M) = \#\text{nonzero}(\sigma_i\text{'s})$

Remark: If M is $n \times n$ and diagonalizable,

$$M = PDP^{-1}$$

where D is diagonal.

Let's compare and contrast

① Every matrix has an SVD. What about an eigendecomp?

No, e.g. $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ need Jordan normal form

② Both can be computed efficiently, e.g. for SVD $\exists$ algs running in time $O(mn^2)$ where $m \geq n$ WLOG

The SVD is widely useful, e.g.

def: $\|M\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} |M_{ij}|^2 \left(= \sum_{i=1}^{r} \sigma_i^2 \right)$

Frobenius norm

Thm [Eckhart-Young]

$$\underbrace{\min_{B,\ \text{rank}(B) \leq k} \|M - B\|_F^2}_{\text{best rank } k \text{ approx}} = \sum_{i=k+1}^{r} \sigma_i^2$$

and is achieved by $B = \underbrace{\sum_{i=1}^{k} \sigma_i u_i v_i^T}_{\text{truncated SVD}}$

This holds for other norms too, e.g.

def: $\|M\|_{op} = \max_{v, \|v\|=1} \|Mv\|$

operator norm

$$= \max_{u, \|u\|=1} \|u^T M\|$$

(update E-Y thm by replacing F-subscripts with op)

Check: If $k = \text{rank}(M)$ then best rank k approx is M itself

---

Thus SVD $\Rightarrow$ best rank k approx

What else can you do with the SVD?

Setup: $\overset{m \times n}{M} \Rightarrow$ distribution on n-dimensional vectors via choosing a coln u.a.r.

Further sps $\mathbb{E}[x] = 0$

thm [PCA]

$$\underbrace{\underset{\substack{P \text{ is proj. onto} \\ k\text{-dimensional subspace}}}{\arg\max} \mathbb{E}[\|Px\|_2^2]}_{\text{max projected variance}}$$

is achieved by $\underbrace{U_{1:k}}_{\text{first } k \text{ colns of } U,\ \text{assuming } \sigma_1 \geq \sigma_2 \geq \dots} U_{1:k}^T$

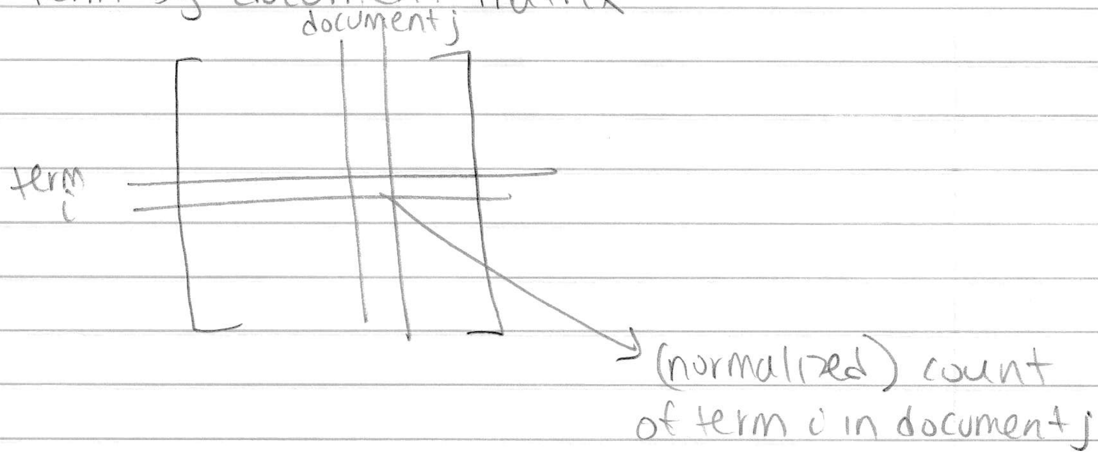
SVD $\Rightarrow$ reduce dimension while maximizing projected variance

---

<u>Latent Semantic Indexing (LSI)</u>

[Deerwester et al]

term-by-document matrix



How can we measure similarity btwn two documents?

<u>Naive</u>: $M_i^T M_j = \sum_{\text{terms}} \begin{pmatrix}\text{normalized} \\ \text{count in} \\ \text{document } i\end{pmatrix} \begin{pmatrix}\text{normalized} \\ \text{count in} \\ \text{document } j\end{pmatrix}$

<u>Problem</u> Documents are sparse

LSI $M_i^T U_{1:k} U_{1:k}^T M_j$

(Sometimes use Σ^{-1} too)

Hope: map documents to "topic" space and compute inner-product there

e.g. what if topics are disjoint collections of words?

[Papadimitriou et al] showed LSI works in this setting

We have techniques that work (somewhat) and we can analyze/justify them

---

Failings of LSI

1. "topics" are orthogonal

   e.g. politics vs. finance

2. "topics" contain negative words

[Lawton, Sylvestre] [Yannakakis]

def: [Hofman], [Lee, Seung] A Nonnegative Matrix Factorization (NMF) of inner-dim r is a decomposition

$$\underset{m\times n}{M} = \underset{m\times r}{A}\,\underset{r\times n}{W}$$

where $A, W \geq 0$

Moreover $\text{rank}^+(M)$ = minimum r s.t. such a decomp exists

Let's interpret NMF

Claim: Sps $M \geq 0$ and its colns sum to one. Then WLOG

$$M = AW$$

colns also sum to one

Proof: Sps $M = \tilde{A}\tilde{W}$ is an NMF. Let

$$D_{ii} = \sum_{j=1}^{m} \tilde{A}_{ij} \text{ and}$$

$$M = \underbrace{\tilde{A} D^{-1}}_{A} \ \underbrace{D \tilde{W}}_{W} \qquad \boxed{}$$

Thus we have

colns of $A \leftrightarrow$ topics $\triangleq$ distributions on words

colns of $W \leftrightarrow$ distributions on topics, i.e. composition of docs

---

Are there efficient algorithms for NMF?

in the worst-case

Meta thm: In ML, usually not

Thm [Vavasis] Computing $\mathrm{rank}^+(M)$ is NP-hard

Actually its $\exists\mathbb{R}$-hard

Goal: What makes ML tractable?

Why do heuristics seem to work so well?
What makes an ML problem well posed?

Natural assumption

def: [Donoho, Stodden] A is separable if for every column j, $\exists$ row i where $A_{ij} > 0$ but $A_{ij'} = 0 \; \forall j' \neq j$

Think about "personal finance"

(0.15, monopoly), (0.09, risk), (0.08, retire), ...

these words can occur in other contexts

But what about "401k"? We call this an anchor word

Thm [Arora, Ge, Kannan, Moitra] There is a polynomial time algorithm to solve separable NMF

Runs in time $O(mnr + mr^{3.5})$